Desarrollando una arquitectura de microservicios mediante MDE

Santiago Meliá[1], Jesús M. Hermida[2], Cristina Cachero[1] y Jaume Aragonés[1]

[1]DLSI. Universidad de Alicante, Spain
{santi, ccachero, jaume}@dlsi.ua.es
[2]European Commission Joint Research Centre, Ispra, Italy
jesus.hermida@ec.europa.eu

Resumen. En los últimos años, la industria del software ha apostado por la migración hacia las aplicaciones basadas en servicios y su despliegue en la nube por su promesa de obtener alta disponibilidad y escalabilidad. Tanto las aplicaciones Web como las móviles utilizan partes servidoras basadas en fachadas REST o SOA que en muchas ocasiones crecen tanto a nivel de servicios como de datos lo que complica su mantenibilidad. En este sentido, ha aparecido recientemente un estilo arquitectónico denominado microservicios que propone la división horizontal de la funcionalidad de una aplicación en una colección de servicios que gestionan separadamente su propia lógica y sus datos. Esta división permite explotar la escalabilidad de la nube a nivel de servicio y abordar los cambios más rápidamente. A pesar de sus beneficios, este estilo arquitectónico presenta algunas desventajas como la dificultad de agregar datos de diferentes microservicios y el mantenimiento de la consistencia entre las diferentes orígenes de datos.

Para abordar estos dos retos, este trabajo presenta una solución MDE basada en una evolución del modelo de servicios de OOH4RIA. Este modelo permite tanto acelerar la creación de microservicios como facilitar el mantenimiento en la comunicación y la composición de datos de diferentes orígenes.

1 Introducción

En los últimos años, la industria del software ha apostado por la migrar sus aplicaciones a arquitecturas basadas en servicios y por su despliegue en la nube por la promesa de obtener una mayor disponibilidad y escalabilidad. Se definen así aplicaciones que por un lado tienen el código de una interfaz de usuario (IU) Web o móvil, y por otro, el código de parte servidora que proporciona la funcionalidad a través de servicios REST o SOA.

En este contexto, aparece un nuevo estilo arquitectónico denominado Microservicios [1] que persigue implementar un sistema software mediante un conjunto de pequeños servicios independientes y autocontenidos, que pueden implementarse en tecnologías diferentes, desplegarse en plataformas distintas y solo necesitan comuni-

adfa, p. 1, 2011.

carse entre sí en casos excepcionales mediante llamadas ligeras REST. Esta arquitectura, avalada por casos de éxito muy relevantes como son Netflix, Ebay o Amazon, ha dejado de relieve su capacidad para abordar grandes proyectos.

Sin embargo, no todos los aspectos de los microservicios son positivos, existen algunas desventajas. (1) Debido a que cada microservicio tiene su propia fuente de datos independiente es más difícil de realizar consultas que agreguen datos de diferentes microservicios. Por otro lado, (2) al poder utilizar fuentes de datos heterogéneas (relacionales, NoSQL, etc.) se descarta el uso de transacciones distribuidas para mantener la integridad de los datos. Asimismo, (3) al tratarse de una arquitectura más compleja que la monolítica, en muchas ocasiones se descarta como solución cuando lo que prima es la puesta rápida en el mercado.

Para resolver estos problemas, se presenta en este trabajo, una propuesta basada en la aplicación del desarrollo dirigido por modelos para aumentar la productividad en la fase de desarrollo de microservicios y reducir la curva de aprendizaje de esta arquitectura. De forma más específica, este trabajo presenta una extensión del modelo de servicios de OOH4RIA [2,5] para la definición de microservicios y la comunicación entre los mismos. Para ello, se propone representar con dicho modelo peticiones dirigidas por eventos que permitan coordinar las actualizaciones de los datos cuando sea necesario.

Para darle soporte a la solución, actualmente se está desarrollado la propuesta dentro de la herramienta OIDE [3] que proporciona los editores de los modelos y el las transformaciones modelo a texto para generar el código completo de cada microservicio en la tecnología .NET (WebAPI para REST, WCF para SOA y NHibernate e EntityFramework para la lógica de negocio y persistencia).

2 Modelando los microservicios

Según Richardson y Smith [4], el desarrollo de microservicios tiene dos retos fundamentales: a) realizar consultas que agreguen datos de diferentes microservicios, y b) mantener la consistencia de los datos entre los microservicios por medio de actualizaciones.

Uno de los aspectos que resulta más atractivo del estilo arquitectónico de microservicios es la división horizontal de la funcionalidad que permite que pequeños equipos trabajen concurrentemente para desarrollar cada microservicio. En este sentido, se ha considerado fundamental que propuesta ayude a preservar esta forma de trabajo. Para ello, se plantea utilizar dos modelos de OOH4RIA independientes de plataforma (PIM): 1) el modelo de dominio [2] para definir la lógica de negocio y su persistencia de datos, en este sentido es equivalente a servicio convencional; y 2) una extensión del modelo de servicios [5] para definir la fachada del microservicio (REST o SOA), y la comunicación entre ellos.

Para mostrar las aportaciones utilizaremos como ejemplo el desarrollo de una aplicación de gestión de pedidos, basada en el ejemplo de [4]. Para ello, en este artículo utilizaremos la notación textual de ambos modelos OOH4RIA.

2.1 Agregación de datos con el modelo de servicios OOH4RIA

Siguiendo el principio de composición de datos definido en [4], el modelo de servicios de OOH4RIA permite definir nuevas estructuras en las clases de servicio (ServiceClass) con atributos propios y atributos que provienen de otras clases de servicio, mediante el patrón Transfer Object Assembler. En el caso de los microservicios, se realiza también siguiendo este patrón.

```
1  ServiceModel EnviosAPI [GestionEnvios]{
2    ServiceClass Pedido [PedidoMS.Pedido] {
3     attributes {
4       numero [num];
5       descripcion [descr];
6       nombreCliente  [-> ClientesAPI.Cliente.nombre] :
7  String;
8     }
9     operations {
10      crearPedido[new (p_cliente:Integer,
11 p_details:List<OrderLine>)]:void
12      abrirPedido[modify (p_id:Integer)]:void
13    }
14    events {
15      define pedidoCreado;
16      notify crearPedido -> pedidoCreado;
17      subscribe ClienteAPI.Cliente.creditoReservado->
18 abrirPedido;
19    }
20 }}
```

```
ServiceModel ClienteAPI [GestionClientes]{
  ServiceClass Cliente [ClienteMS.Cliente] {
   attributes {
      nombre [nombre];
      credito [credito];
   }
   operations {
reservarCredito[modify(p_customer_oid:Integer)]:
Boolean;
   }
   events
   {
     define creditoReservado;
     subscribe EnviosAPI.Pedido.pedidoCreado->
reservarCredito;
     notify reservarCredito -> creditoReservado;
   }
}}
```

Figura 1. Fragmentos del modelo de servicios de OOH4RIA para un sistema de gestión de pedidos.

En nuestra propuesta, el modelo de servicio OOH4RIA ha sido extendido incorporando la directiva `[->]` para incluir atributos procedentes de otras clases de servicio. En la figura 1, en la que se han definido los microservicios EnviosAPI y ClientesAPI, la clase de servicio Pedido (EnviosAPI) tiene un atributo llamado *nombreCliente* cuyo valor procede de la clase de servicio Cliente del microservicio ClientesAPI (líneas 6-7). De esta forma, se generará una clase *assembler* que, cuando construye los datos del pedido, recupera también el nombre del cliente realizando la invocación al otro microservicio.

2.2 Consistencia entre microservicios mediante una arquitectura orientada a eventos

Los datos de un microservicio son locales al mismo, son solo accesibles mediante el API y pueden almacenarse en fuentes heterogéneas. Por tanto, es imposible utilizar transacciones distribuidas para mantener su consistencia. La solución pasa por utilizar una arquitectura dirigida por eventos, donde un microservicio publica un evento solo cuando algo significativo haya sucedido, entonces el resto de microservicios que estén suscritos reciben la notificación y toman alguna medida.

Con OOH4RIA podemos definir una serie de eventos asociados a los microservicios y especificar como se utilizan para mantener la consistencia de los datos de nuestra aplicación. En nuestra propuesta, el modelo de servicio OOH4RIA ha sido extendido con tres nuevas directivas para poder definir eventos y su utilización en cada clase de servicio en el apartado *events*: 1) *define*: crea un nuevo evento dependiente de la clase de servicio; 2) *subscribe*: hace que la clase de servicio escuche un determinado evento y lance una operación si el evento se publica y 3) *notify*: publica un evento cuando finaliza la ejecución de una operación.

En el ejemplo, en el microservicio *EnviosAPI* se ha definido (*define*, línea 15) un evento llamado *pedidoCreado* que se publica (*notify*, línea 16) al finalizar la operación crearPedido (servicio Pedido). En ese momento, el servicio *Cliente* del microservicio *ClienteAPI* que estaba suscrito (*subscribe,* líneas 14-15) previamente a dicho evento, lo recibe y ejecuta la operación *reservarCrédito* para comprobar si el cliente tiene crédito. Solo si el cliente tiene crédito, el microservicio *ClienteAPI* publica el evento *creditoReservado* (línea 16). El servicio Pedido del microservicio EnviosAPI, al estar también suscrito a este segundo evento (líneas 17-18), ejecuta la operación *abrirPedido* que prosigue con el proceso de pedido.

Agradecimientos: Este estudio ha sido financiado por el proyecto TIN2016-78103-C2-2-R: "Plataforma para la publicación y consumo de datos abiertos para una ciudad inteligente" del Ministerio de Economía, Industria y Competitividad.

Referencias

1. Fowler, M., Lewis, J.: Microservices. http://martinfowler.com/articles/microservices.html. Accessed 15 Jun 2015.
2. Meliá, S., Gomez, J., Perez, S., & Diaz, O. A model-driven development for GWT-based Rich Internet Applications with OOH4RIA. *ICWE 2008*. (pp. 13-23).
3. Meliá, S., Martínez, J. J., Mira, S., Osuna, J. A., & Gómez, J. *An eclipse plug-in for model-driven development of rich internet applications* (pp. 514-517). ICWE 2010.
4. Richardson, C., Smith F.: Microservice: From Design to Deployment. Free book, NGINX, Inc, 2016.
5. Arias, A., Hermida, J.M., Meliá, S. Desarrollo de una fachada de servicios REST/SOA para aplicaciones SOFEA mediante una aproximación MDE (pp. 515-518). JISBD 2016.